Compound Term Composition Algebra: The Semantics

Tzitzikas, Yannis; Analti, Anastasia; Spyratos, Nicolas

Published in:
Journal on Data Semantics

Publication date:
2004

Compound Term Composition Algebra: The Semantics

Yannis Tzitzikas[1,4], Anastasia Analyti[2], Nicolas Spyratos[3]

[1] Institut d'Informatique , F.U.N.D.P. (University of Namur), Belgium
[2] Institute of Computer Science, FORTH, Heraklion, Greece
[3] Laboratoire de Recherche en Informatique, Universite de Paris-Sud, France
Email : ytz@info.fundp.ac.be, analyti@ics.forth.gr, spyratos@lri.fr

Abstract. The *Compound Term Composition Algebra* (CTCA) is an algebra with four algebraic operators, whose composition can be used to specify the meaningful (valid) compound terms (conjunctions of terms) in a given faceted taxonomy in an efficient and flexible manner. The "positive" operations allow the derivation of valid compound terms through the declaration of a small set of valid compound terms. The "negative" operations allow the derivation of valid compound terms through the declaration of a small set of invalid compound terms. In this paper, we formally define the model-theoretic semantics of the operations and the closed-world assumptions adopted in each operation. We prove that CTCA is monotonic with respect to both valid and invalid compound terms, meaning that the valid and invalid compound terms of a subexpression are not invalidated by a larger expression. We show that CTCA cannot be directly represented in Description Logics. However, we show how we could design a metasystem on top of Description Logics in order to implement this algebra.

Keywords: Faceted Taxonomies, Semantics, Description Logics.

1 Introduction

A *faceted taxonomy* is a set of taxonomies, each describing a given domain from a different aspect, or *facet* (for more about faceted classification and analysis see [12, 6, 18, 7, 9, 10, 8]). Having a faceted taxonomy, the indexing of domain objects is done through conjunctive combinations of terms from the facets, called *compound terms*. Faceted taxonomies are used in Web Catalogs [11], Libraries [8], Software Repositories [9, 10], and several others application domains. Current interest in faceted taxonomies is also indicated by several recent or ongoing

[4] Part of this work was done while the author was an ERCIM fellow at the VTT Technical Research Centre of Finland.

projects (like FATKS[5], FACET[6], FLAMENGO[7]) and the emergence of XFML [1] (Core-eXchangeable Faceted Metadata Language) a markup language for applying the faceted classification paradigm on the Web.

For example, assume that the domain of interest is a set of hotel Web pages in Greece, and suppose that we want to provide access to these pages according to the *Location* of the hotels and the *Sports* facilities they offer. Figure 1 shows these two facets. Each object is described using a *compound term*. For example, a hotel in Crete providing sea ski and wind-surfing facilities would be described by the compound term $\{Crete, SeaSki, Windsurfing\}$.

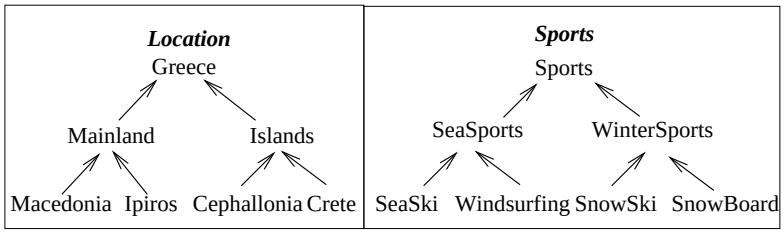

Fig. 1. Two facets for indexing hotel Web pages

Faceted taxonomies carry a number of well known advantages over single hierarchies in terms of building and maintaining them, as well as using them in multicriteria indexing. For instance, assume that the Web consists of 1 billion pages and suppose we want to create terms that allow partitioning the pages of the Web in blocks of 10 pages as it is illustrated in Figure 2. For doing so we need at least 100 millions (10^8) different terms, assuming each page is indexed by one term. If we want these terms to be the leaves of a complete balanced decimal tree, then this tree would have: 111,111,111 terms in total. By adopting a faceted taxonomy we can obtain the same discrimination capability with much fewer terms. For example, consider 4 facets, each one having 100 leaf terms. The number of all combinations of these leaf terms, with one term from each facet, equals 100 millions. If each facet is a complete balanced decimal tree, then the entire faceted taxonomy would have: (100 + 10 + 1) x 4 = 444 terms in total. We can obtain the same discrimination capability with even fewer terms! For example, we can have 10^8 different combinations by adopting 8 facets, each one having 10 leaf terms. In this case, the entire faceted taxonomy has only 88 terms! Notice the tremendous difference between 111,111,111 and 88. It is therefore evident that a faceted taxonomy has several advantages by comparison to a single taxonomy (of the kind of Yahoo! or ODP), such as conceptual clarity, compactness and scalability (e.g. see [10]). A drawback, however, is the cost of avoiding *invalid* combinations, i.e. compound terms that do not apply to any object in the domain. For example, the compound term $\{Crete, SnowBoard\}$ is

[5] http://www.ucl.ac.uk/fatks/database.htm
[6] http://www.glam.ac.uk/soc/research/hypermedia/facet_proj/index.php
[7] http://bailando.sims.berkeley.edu/flamenco.html

an invalid compound term, as there are no hotels in Crete offering snow-board facilities (because Crete never has enough snow). These meaningless or invalid compound terms may give rise to problems and errors during object indexing.

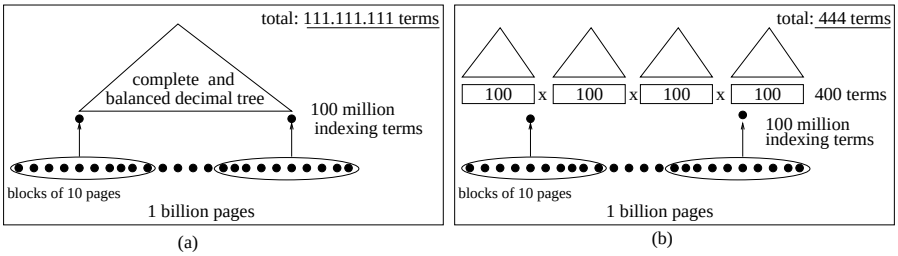

Fig. 2. The benefits of using faceted instead of non-faceted taxonomies

In [13], we proposed the *Compound Term Composition Algebra* (CTCA), an algebra that allows the efficient and flexible specification of the valid compound terms over a faceted taxonomy. Having defined the set of valid compound terms, a navigation tree can be derived dynamically, whose nodes correspond to valid compound terms, only. Such a navigation tree can aid object indexing and browsing, and can prevent some of the indexing errors that may occur in an open and collaborative environment like the Web. Following this approach, given a faceted taxonomy, one can use an *algebraic expression* to define the desired set of compound terms. In each algebraic operation, the designer has to declare either a small set of valid compound terms from which other valid compound terms are inferred, or a small set of invalid compound terms from which other invalid compound terms are inferred. Then, a *closed-world assumption* is adopted for the rest of the compound terms in the range of the operation. For example, if a user declares in a positive operation that the compound term $\{Crete, SeaSki\}$ is valid then it is inferred that the compound term $\{Crete, SeaSports\}$ is also valid. If a user declares in a negative operation that the compound term $\{Crete, WinterSports\}$ is invalid then it is inferred that the compound term $\{Crete, SnowBoard\}$ is also invalid. In our example, this means that the designer can specify all valid compound terms of the faceted taxonomy by providing a relatively small number of (valid or invalid) compound terms. This is an important feature as it minimizes the effort needed by the designer.

From an application point of view, an important remark is that there is no need to store the set of valid compound terms that are defined by an expression, as an inference mechanism (given in [13]) can check whether a compound term belongs to the set of compound terms defined by an expression in polynomial time. So, only the faceted taxonomy and the expression have to be stored. Another final remark, is that the recently emerged markup language XFML+CAMEL (*C*ompound term composition *A*lgebraically-*M*otivated *E*xpression *L*anguage) [2], allows publishing and exchanging faceted taxonomies *and* expressions of CTCA in an XML format. An authoring system based on

CTCA has just been developed by VTT and Helsinki University of Technology (HUT), under the name FASTAXON [14].

In this paper, we emphasize on the *semantics* of the algebra. Specifically, we formally define the model-theoretic semantics of the operations and the closed-world assumptions adopted in each operation. First, intermediate semantics are defined for the particular operations, and then intermediate semantics are synthesized to define the semantics of the complete algebraic operation. Based on these, we define the models of an algebraic expression, and we prove that every *well-formed* algebraic expression is satisfiable. We also prove that CTCA is *monotonic* with respect to both valid and invalid compound terms, meaning that the valid and invalid compound terms of a subexpression are not invalidated by a larger expression. The importance of this property is demonstrated through an example.

We also show that CTCA cannot be directly represented in Description Logics. However we show how a meta-system (on top of a Description Logics-based system) could be designed in order to implement CTCA.

The remaining of this paper is organized as follows: Section 2 describes the algebra, and justifies the definition of a well-formed algebraic expression based on the monotonicity property. Section 3 defines the model-theoretic semantics of the algebra and proves monotonicity. Section 4 compares the approach with Description Logics. Finally, Section 5 concludes the paper and discusses applications. Proofs of all propositions are given in Appendix A. Appendix B illustrates the application of the algebra and the benefits of its monotonic nature by an example. A table of symbols is given in Appendix C.

2 The Compound Term Composition Algebra

In this section, we present in brief the *Compound Term Composition Algebra*, defined in [13]. For more explanations, and examples the reader should refer to that article.

A *terminology* is a finite set of names, called *terms*. A *taxonomy* is a pair $(\mathcal{T}, \leq)$, where $\mathcal{T}$ is a *terminology* and $\leq$ is a reflexive and transitive relation over $\mathcal{T}$, called *subsumption*.

A *compound term* over $\mathcal{T}$ is any subset of $\mathcal{T}$. For example, the following sets of terms are compound terms over the terminology *Sports* of Figure 1: $s_1 = \{SeaSki, Windsurfing\}$, $s_2 = \{SeaSports\}$, and $s_3 = \emptyset$.

A *compound terminology* S over $\mathcal{T}$ is any set of compound terms that contains the compound term $\emptyset$.

The set of all compound terms over $\mathcal{T}$ can be ordered using the *compound ordering* over $\mathcal{T}$, defined as: $s \preceq s'$ iff $\forall t' \in s' \ \exists t \in s$ such that $t \leq t'$.

That is, $s \preceq s'$ iff s contains a narrower term for every term of s'. In addition, s may contain terms not present in s'. Roughly, $s \preceq s'$ means that s carries more specific information than s'. For example, $\{SeaSki, Windsurfing\} \preceq \{SeaSports\} \preceq \emptyset$.

We say that two compound terms s, s' are *equivalent* iff $s \preceq s'$ and $s' \preceq s$. For example, $\{SeaSki, SeaSports\}$ and $\{SeaSki\}$ are equivalent. Intuitively, equivalent compound terms carry the same information.

Definition 1. A *compound taxonomy* over $\mathcal{T}$ is a pair $C = (S, \preceq)$, where S is a compound terminology over $\mathcal{T}$, and $\preceq$ is the compound ordering over $\mathcal{T}$ restricted to S.

Let $P(\mathcal{T})$ be the set of all compound terms over $\mathcal{T}$ (i.e. the powerset of $\mathcal{T}$). Clearly, $(P(\mathcal{T}), \preceq)$ is a compound taxonomy over $\mathcal{T}$.

Let s be a compound term. The broader and the narrower compound terms of s are defined as follows:

$$\mathrm{Br}(s) = \{s' \in P(\mathcal{T}) \mid s \preceq s'\}$$
$$\mathrm{Nr}(s) = \{s' \in P(\mathcal{T}) \mid s' \preceq s\}$$

Let S be a compound terminology over $\mathcal{T}$. The broader and the narrower compound terms of S are defined as follows:

$$Br(S) = \cup\{\mathrm{Br}(s) \mid s \in S\}$$
$$Nr(S) = \cup\{\mathrm{Nr}(s) \mid s \in S\}$$

One way of designing a taxonomy is by identifying a number of different aspects of the domain of interest and then designing one taxonomy per aspect. As a result we obtain a set of taxonomies called *facets*. Given a set of facets we can define a *faceted taxonomy*.

Definition 2. Let $\{F_1, ..., F_k\}$ be a finite set of taxonomies, where $F_i = (\mathcal{T}_i, \leq_i)$, and assume that the terminologies $\mathcal{T}_1$, ... ,$\mathcal{T}_k$ are pairwise disjoint. Then the pair $\mathcal{F} = (\mathcal{T}, \leq)$, where

$$\mathcal{T} = \bigcup_{i=1}^{k} \mathcal{T}_i \quad \text{and} \quad \leq \; = \bigcup_{i=1}^{k} \leq_i,$$

is a taxonomy which we shall call the *faceted taxonomy generated* by $\{F_1, ..., F_k\}$. We shall call the taxonomies $F_1, ..., F_k$ the *facets* of $\mathcal{F}$.

Clearly, all definitions introduced so far apply also to $(\mathcal{T}, \leq)$. For example, the set $S = \{\{Greece\}, \{Sports\}, \{SeaSports\}, \{Greece, Sports\}, \{Greece, SeaSports\}, \emptyset\}$ is a compound terminology over the terminology $\mathcal{T}$ of the faceted taxonomy shown in Figure 1. Additionally, the pair $(S, \preceq)$ is a compound taxonomy over $\mathcal{T}$.

Let $\mathcal{F} = (\mathcal{T}, \leq)$ be the faceted taxonomy generated by a given set of facets $\{F_1, ..., F_k\}$. The problem is that $\mathcal{F}$ does not itself specify which compound terms, i.e. which elements of $P(\mathcal{T})$, are valid (i.e. meaningful) and which are not (i.e. meaningless). To tackle this problem, we introduce an algebra for defining a compound terminology over $\mathcal{T}$ (i.e. a subset of $P(\mathcal{T})$) which consists of the valid compound terms.

2.1 Algebraic operations

For defining the desired compound taxonomy the designer has to formulate an algebraic expression e, using four operations, namely:

- *plus-product*,
- *minus-product*,
- *plus-self-product*, and
- *minus-self-product*.

Let us now see which are the initial operands of these operations. To each facet terminology $\mathcal{T}_i$ we associate a compound terminology, denoted by T_i, that we call the *basic compound terminology* of $\mathcal{T}_i$, given by:

$$T_i = \cup\{\ \mathrm{Br}(\{t\}) \mid t \in \mathcal{T}_i\} \tag{1}$$

So the initial operands (or "building blocks") of the algebraic operations are the basic compound terminologies $\{T_1, .., T_k\}$. Let us now explain the role of "Br" in the formula (1). We by default assume that every individual term of a taxonomy is valid (meaningful), i.e. there are real-world objects (at least one) to which this term applies. It follows easily that all compound terms in $\mathrm{Br}(\{t\})$ are valid too. We used $\mathrm{Br}(\{t\})$ instead of just $\{t\}$ in order to capture the case where $\mathcal{T}_i$ is not a tree. For example, suppose three terms a, b and c such that $a \leq b$ and $a \leq c$. It follows that the compound term $\{b, c\}$ is certainly valid as it subsumes $\{a\}$. Formula (1) captures this case, as $\{b, c\} \in \mathrm{Br}(\{a\})$. Of course, if $\mathcal{T}_i$ had a tree structure then we could omit "Br" and rewrite formula (1) as: $T_i = \cup\{\ \{t\} \mid t \in \mathcal{T}_i\} \cup \{\emptyset\}$.

Let $\mathcal{S}$ be the set of all compound terminologies over $\mathcal{T}$. Before defining the four algebraic operations, we shall first define an auxiliary n-ary operation $\oplus$ over $\mathcal{S}$, called *product*. This operation results in an "unqualified" compound terminology whose compound terms are *all* possible combinations of compound terms from its operands. Specifically, if $S_1, ..., S_n$ are compound terminologies, then:

$$S_1 \oplus ... \oplus S_n = \{s_1 \cup ... \cup s_n \mid s_i \in S_i\}$$

Now *plus-product* and *minus-product* are two "variations" of the $\oplus$ operation. Each of these two operations has an extra parameter denoted by P or N, respectively. The set P is a set of compound terms that the designer considers as valid. On the other hand, the set N is a set of compound terms that the designer considers as invalid. These parameters are declared by the designer (domain expert).

To proceed and explain the role of these parameters we need to distinguish what we shall call *genuine compound terms*. Intuitively, a genuine compound term combines non-empty compound terms from more than one compound terminology.

Definition 3. The set of *genuine* compound terms over a set of compound terminologies $S_1, ..., S_n$, denoted by $G_{S_1,...,S_n}$, is defined as follows:

$$G_{S_1,...,S_n} = S_1 \oplus ... \oplus S_n - \bigcup_{i=1}^{n} S_i$$

For example if
$S_1 = \{\{Greece\}, \{Islands\}, \emptyset\}$,
$S_2 = \{\{Sports\}, \{WinterSports\}, \emptyset\}$, and
$S_3 = \{\{Pensions\}, \{Hotels\}, \emptyset\}$, then

$$\{Greece, WinterSports, Hotels\} \in G_{S_1,S_2,S_3},$$
$$\{WinterSports, Hotels\} \in G_{S_1,S_2,S_3} \text{ , but}$$
$$\{Hotels\} \notin G_{S_1,S_2,S_3}$$

One can easily see, that as we are interested in characterizing the validity of *compound terms*, the parameters P and N must contain *genuine* compound terms only.

We can now define precisely the *plus-product* operation, $\oplus_P$.

Definition 4. Let $S_1, ..., S_n$ be compound terminologies and $P \subseteq G_{S_1,...,S_n}$. The *plus-product* of $S_1, ..., S_n$ with respect to P, denoted by $\oplus_P(S_1, ..., S_n)$, is defined as follows:

$$\oplus_P(S_1, ...S_n) = S_1 \cup ... \cup S_n \ \cup \ Br(P)$$

This operation results in a compound terminology consisting of the compound terms of the initial compound terminologies, *plus* the compound terms which are broader than an element of P. This is because, if a compound term p is valid then all compound terms in $\mathrm{Br}(p)$ are also valid.

For any parameter P, it holds: $\bigcup_{i=1}^{n} S_i \subseteq \oplus_P(S_1, ..., S_n) \subseteq S_1 \oplus ... \oplus S_n$.

Let us now define precisely the *minus-product* operation, $\ominus_N$.

Definition 5. Let $S_1, ..., S_n$ be compound taxonomies and $N \subseteq G_{S_1,...,S_n}$. The *minus-product* of $S_1, ..., S_n$ with respect to N, denoted by $\ominus_N(S_1, ..., S_n)$, is defined as follows:

$$\ominus_N(S_1, ...S_n) = S_1 \oplus ... \oplus S_n - Nr(N)$$

This operation results in a compound terminology consisting of all compound terms in the product of the initial compound terminologies, *minus* all compound terms which are narrower than an element of N. This is because, if a compound term n is invalid then every compound term in $\mathrm{Nr}(n)$ is invalid.

For any parameter N, it holds: $\bigcup_{i=1}^{n} S_i \subseteq \ominus_N(S_1, ..., S_n) \subseteq S_1 \oplus ... \oplus S_n$.

For example, consider the compound terminologies S and S' shown in the left part of Figure 3, and suppose that we want to define a compound terminology that does not contain the compound terms $\{Islands, WinterSports\}$ and

$\{Islands, SnowSki\}$, because they are invalid. For this purpose, we can use either a *plus-product* or a *minus-product* operation.

Specifically, we can use a plus-product operation, $\oplus_P(S, S')$, where $P = \{\{Islands, Seasports\}, \{Greece, SnowSki\}\}$. The compound taxonomy defined by this operation is shown in the right part of Figure 3. In this figure we enclose in squares the elements of P. We see that the compound terminology $\oplus_P(S, S')$ contains the compound term $s = \{Greece, Sports\}$, as $s \in Br(\{Islands, SeaSports\})$. However, it does not contain the compound terms $\{Islands, WinterSports\}$ and $\{Islands, SnowSki\}$, as they do not belong to $S \cup S' \cup Br(P)$.

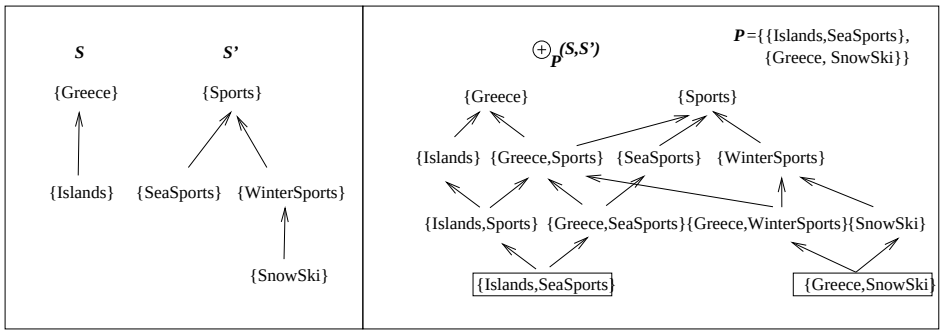

Fig. 3. An example of a *plus-product*, $\oplus_P$, operation

Alternatively, we can obtain the compound taxonomy shown at the right part of Figure 3 by using a *minus-product* operation, i.e. $\ominus_N(S, S')$, with $N = \{\{Islands, WinterSports\}\}$. The result does not contain the compound terms $\{Islands, WinterSports\}$ and $\{Islands, SnowSki\}$, as they are elements of $Nr(N)$.

The two operations introduced so far allow defining a compound terminology which consists of compound terms that contain at most one compound term from each basic compound terminology. However, in general there may exist valid compound terms that contain more than one term from the same facet (multiple classification within one facet). To capture such cases, and specify which of these compound terms are valid and which are not, the algebra supports another two operations, namely, *plus-self-product* and *minus-self-product*.

Again, we shall start from an auxiliary operation called *self-product*. *Self-product*, $\overset{*}{\oplus}$, is a unary operation which gives all possible compound terms of one facet. The *self-product* of T_i is defined as: $\overset{*}{\oplus}(T_i) = P(\mathcal{T}_i)$.

Now *plus-self-product* and *minus-self-product* are two "variations" of the $\overset{*}{\oplus}$ self-product operation. Each of these two operations has an extra parameter denoted by P or N, respectively. Again, the notion of genuine compound terms is also necessary here. The set of *genuine* compound terms over a basic compound

terminology T_i is defined as:

$$G_{T_i} = \overset{*}{\oplus}(T_i) - T_i$$

Now we define precisely the *plus-self-product* operation, $\overset{*}{\oplus}_P$.

Definition 6. Let T_i be a basic compound terminology and $P \subseteq G_{T_i}$. The *plus-self-product* of T_i with respect to P, denoted by $\overset{*}{\oplus}_P(T_i)$, is defined as follows:

$$\overset{*}{\oplus}_P(T_i) = T_i \cup Br(P)$$

This operation results in a compound terminology consisting of the compound terms of the initial basic compound terminology, *plus* all compound terms which are broader than an element of P.

For any parameter P, it holds: $T_i \subseteq \overset{*}{\oplus}_P(T_i) \subseteq \overset{*}{\oplus}(T_i)$

Now *minus-self-product* operation, $\overset{*}{\ominus}_N$, is defined as:

Definition 7. Let T_i be a basic compound terminology and $N \subseteq G_{T_i}$. The *minus-self-product* of T_i with respect to N, denoted by $\overset{*}{\ominus}_N(T_i)$, is defined as follows:

$$\overset{*}{\ominus}_N(T_i) = \overset{*}{\oplus}(T_i) - Nr(N)$$

This operation results in a compound terminology consisting of all compound terms in the self-product of T_i, *minus* the compound terms which are narrower than an element in N.

For any parameter N it holds: $T_i \subseteq \overset{*}{\ominus}_N(T_i) \subseteq \overset{*}{\oplus} T_i$

Table 1 gives the definition of each operation of the algebra.

Operation	e	S_e	arity
product	$S_1 \oplus ... \oplus S_n$	$\{ s_1 \cup ... \cup s_n \mid s_i \in S_i\}$	n-ary
plus-product	$\oplus_P(S_1, ...S_n)$	$S_1 \cup ... \cup S_n \cup Br(P)$	n-ary
minus-product	$\ominus_N(S_1, ...S_n)$	$S_1 \oplus ... \oplus S_n - Nr(N)$	n-ary
self-product	$\overset{*}{\oplus}(T_i)$	$P(\mathcal{T}_i)$	unary
plus-self-product	$\overset{*}{\oplus}_P(T_i)$	$T_i \cup Br(P)$	unary
minus-self-product	$\overset{*}{\ominus}_N(T_i)$	$\overset{*}{\oplus}(T_i) - Nr(N)$	unary

Table 1. The operations of the Compound Term Composition Algebra

2.2 Algebraic Expressions

For defining the desired compound taxonomy, the designer has to formulate an expression e, where an expression is defined as follows:

Definition 8. An expression over a set of facets $\{F_1, ..., F_k\}$ is defined according to the following grammar:

$$e ::= \oplus_P(e, ..., e) \mid \ominus_N (e, ..., e) \mid \overset{*}{\oplus}_P T_i \mid \overset{*}{\ominus}_N T_i \mid T_i$$

The outcome of the evaluation of an expression e is denoted by S_e and is called the *compound terminology* of e. In addition, $(S_e, \preceq)$ is called the *compound taxonomy* of e.

Let $\mathcal{T}_e$ be the union of the terminologies of the facets appearing in an expression e. The expression e actually partitions the set $P(\mathcal{T}_e)$ into two sets:

(a) the set of valid compound terms, S_e, and
(b) the set of invalid compound terms $P(\mathcal{T}_e) - S_e$

Now *well-formed* expressions are defined as follows:

Definition 9. An expression e is *well-formed* iff:

(i) each basic compound terminology T_i appears at most once in e,
(ii) each parameter P that appears in e, is a subset of the associated set of genuine compound terms, e.g. if $e = \oplus_P(e_1, e_2)$ then it should be $P \subseteq G_{S_{e_1}, S_{e_2}}$, and
(iii) each parameter N that appears in e, is also a subset of the associated set of genuine compound terms, e.g. if $e = \overset{*}{\ominus}_N (T_i)$ then it should be $N \subseteq G_{T_i}$.

For example, the expression $(T_1 \oplus_P T_2) \ominus_N T_1$ is not well-formed, as T_1 appears twice in the expression.

Constraints (i), (ii), and (iii) ensure that the evaluation of an expression is *monotonic*, meaning that the valid and invalid compound terms of an expression e increase as the length of e increases[8] (in other words, there are no conflicts). For example, if we omit constraint (i) then an invalid compound term according to an expression $T_1 \oplus_P T_2$ could be valid according to a larger expression $(T_1 \oplus_P T_2) \oplus_{P'} T_1$. If we omit constraint (ii) then an invalid compound term according to an expression $T_1 \oplus_{P_1} T_2$ could be valid according to a larger expression $(T_1 \oplus_{P_1} T_2) \oplus_{P_2} T_3$. Additionally, if we omit constraint (iii) then a valid compound term according to an expression $T_1 \oplus_P T_2$ could be invalid according to a larger expression $(T_1 \oplus_P T_2) \ominus_N T_3$.

This monotonic behaviour in the evaluation of a well-formed expression results in a number of useful properties. Specifically, due to their monotonicity,

[8] Proof of this property is given in Section 3.

well-formed expressions can be formulated in a systematic, gradual manner (intermediate results of subexpressions are not invalidated by larger expressions). Appendix B offers examples of the algebra. The benefits of monotonicity are also demonstrated there.

In the rest of the paper, we assume that expressions are *well-formed*. In [13], we presented the algorithm $IsValid(e, s)$ that checks the validity of a compound term s according to a well-formed expression e in $O(|\mathcal{T}|^2 * |s| * |\mathcal{P} \cup \mathcal{N}|)$ time, where $\mathcal{P}$ denotes the union of all P parameters and $\mathcal{N}$ denotes the union of all N parameters appearing in e. Polynomial is also the time needed for checking if an expression e is well-formed.

Let us define the *size* of an expression e as follows: $size(e) = |\mathcal{P} \cup \mathcal{N}|$. Obviously, reducing the size of e is desirable, as both the storage space requirements of e and the time for checking compound term validity are reduced. It is easy to see that if we replace each parameter P of e with $minimal_{\preceq}(P)$ and each parameter N of e with $maximal_{\preceq}(N)$, we derive an expression e' such that $S_e = S_{e'}$ and $size(e') \leq size(e)$. Yet, e' may not be the shortest expression with these properties. The problem of finding the shortest expression e' such that $S_e = S_{e'}$ is treated in [16].

3 Semantic Interpretation

At first we shall give a model-theoretic interpretation to faceted taxonomies and to compound taxonomies. Using this framework, we shall formally define the validity of a compound term. In the sequent, we will define the models of the compound taxonomies that *satisfy* a well-formed algebraic expression. At that point, it will become evident that the algebraic operations and their parameters actually pose constraints to the models of the compound taxonomy $(P(\mathcal{T}), \preceq)$. Moreover, we will show that the operations as defined in Section 2, are also justified by the semantic interpretation of this section.

We conceptualize the world as a set of objects, that is, we assume an arbitrary domain of discourse and a corresponding set of objects Obj. A typical example of such a domain is a set of Web pages. The only constraint that we impose on the set Obj is that it must be a denumerable set.

The set of objects described by a term is the *interpretation* of that term.

Definition 10. Given a terminology $\mathcal{T}$, we call *interpretation* of $\mathcal{T}$ over Obj any function $I : \mathcal{T} \rightarrow 2^{Obj}$.

Intuitively, the interpretation $I(t)$ of a term t is the set of objects to which the term t is correctly applied. In our discussion the set Obj will be usually understood from the context. So, we shall often say simply "an interpretation" instead of "an interpretation over Obj". Interpretation, as defined above, assigns to a term denotational or extensional meaning [19].

Definition 11. An interpretation I of $\mathcal{T}$ is a *model* of a taxonomy $(\mathcal{T}, \leq)$, if for each $t, t' \in \mathcal{T}$: if $t \leq t'$ then $I(t) \subseteq I(t')$.

Now, any interpretation I of $\mathcal{T}$ can be extended to an interpretation $\hat{I}$ of $P(\mathcal{T})$ as follows:

$$\hat{I}(\{t_1, ..., t_n\}) = I(t_1) \cap I(t_2) \cap ... \cap I(t_n)$$

Definition 12. Let $(\mathcal{T}, \leq)$ be a taxonomy. An interpretation $\hat{I}$ of $P(\mathcal{T})$ is a *model* of $(P(\mathcal{T}), \preceq)$, if for each $s, s' \in P(\mathcal{T})$: if $s \preceq s'$ then $\hat{I}(s) \subseteq \hat{I}(s')$.

Proposition 1. Let S be a compound taxonomy over a taxonomy $\mathcal{T}$, and let s and s' be two elements of S. It holds:

$$s \preceq s' \text{ iff } \hat{I}(s) \subseteq \hat{I}(s') \text{ in every model } I \text{ of } (\mathcal{T}, \leq)$$

We can see that the compound ordering $\preceq$ is also justified semantically (it coincides with extensional subsumption).

From the above, it easily follows that an interpretation I is a model of $(\mathcal{T}, \leq)$ iff $\hat{I}$ is a model of $(P(\mathcal{T}), \preceq)$.

For brevity hereafter we shall denote by I both I and $\hat{I}$. Additionally, in the following, by model I we refer to a model I of $(\mathcal{T}, \leq)$.

For describing the semantics of compound terminologies that are defined by algebraic expressions, we shall equate validity with non-empty interpretation and invalidity with empty interpretation. For simplicity, we consider only expressions of the form $e \oplus_P e'$ and $e \ominus_N e'$, with no plus-self-product and minus-self-product operations. We will define the *valid* and *invalid compound terms* of an expression e, denoted by $VC(e)$ and $IC(e)$, recursively starting by $VC(T_i) = T_i$.

At first, we define the *valid genuine compound terms* of e *op* e' (denoted by $VG(e \text{ } op \text{ } e')$) and the *invalid genuine compound terms* of e *op* e' (denoted by $IG(e \text{ } op \text{ } e')$), based on $VC(e)$ and $VC(e')$. Intuitively, we first define the validity of the genuine compound terms over $VC(e)$ and $VC(e')$.

The valid genuine compound terms of $e \oplus_P e'$ are defined as follows:

$$VG(e \oplus_P e') = \{ \quad s \in G_{VC(e),VC(e')} \mid \\ I(s) \neq \emptyset \text{ in every model } I \text{ such that: } I(s') \neq \emptyset, \forall s' \in P\}$$

Now by adopting a closed-world assumption for the invalid genuine compound terms, we assume that all elements of $G_{S_e,S_{e'}} - VG(e \oplus_P e')$ are invalid. Thus we write:

$$IG(e \oplus_P e') = G_{VC(e),VC(e')} - VG(e \oplus_P e')$$

The following proposition holds:

Proposition 2. $VG(e \oplus_P e') = Br(P) \cap G_{VC(e),VC(e')}$

Below we define the invalid genuine compound terms of $e \ominus_N e'$:

$$IG(e \ominus_N e') = \{ \quad s \in G_{VC(e),VC(e')} \mid \\ I(s) = \emptyset \text{ in every model } I \text{ such that: } \quad I(s') = \emptyset, \forall s' \in N\}$$

Now we again adopt a closed-world assumption for the valid genuine compound terms, specifically we assume that all elements of $G_{VC(e),VC(e')} - IG(e \ominus_N e')$ are valid. Thus we write:

$$VG(e \ominus_N e') = G_{VC(e),VC(e')} - IG(e \ominus_N e')$$

The following proposition holds:

Proposition 3. $IG(e \ominus_N e') = Nr(N) \cap G_{VC(e),VC(e')}$

Until now, for every operation $e \; op \; e'$ we partitioned the set $G_{VC(e),VC(e')}$ to the sets $VG(e \; op \; e')$ and $IG(e \; op \; e')$. Let $\mathcal{T}_e$ denote the union of the terminologies of the facets that appear in e. Now for any well-formed expression e, we will partition the elements of the entire $P(\mathcal{T}_e)$ into the set of *valid compound terms*, $VC(e)$, and the set of *invalid compound terms*, $IC(e)$. We define[9]:

$$VC(T_i) = T_i, \text{ for } i = 1, ...k$$
$$VC(e \; op \; e') = VG(e \; op \; e') \; \cup \; VC(e) \; \cup \; VC(e')$$
$$IC(e) = P(\mathcal{T}_e) - VC(e)$$

Clearly, the sets $VC(e)$ and $IC(e)$ constitute a partition of $P(\mathcal{T}_e)$.

Definition 13. Let e be a well-formed expression, and let I be a model of $(\mathcal{T}, \leq)$. We say that I *satisfies* e if:
(1) $\forall \; s \in VC(e), \; I(s) \neq \emptyset$, and
(2) $\forall \; s \in IC(e), \; I(s) = \emptyset$.

The following proposition expresses that every expression e is satisfiable.

Proposition 4. Let e be a well-formed expression. There always exists a model I of $(\mathcal{T}, \leq)$ that satisfies e.

The following proposition expresses that the compound taxonomy S_e of an algebraic expression e (as computed from our operations) consists of exactly those compound terms which are valid according to the semantic interpretation that we described in this section.

[9] Note that in the definition, there is double recursion.

Proposition 5. Let e be a well-formed expression. It holds:

$$VC(e) = S_e \text{ and } IC(e) = P(\mathcal{T}_e) - S_e$$

The following proposition gives a very important property of our theory, that is, intermediate results of subexpressions are not invalidated by larger expressions. Thus, expressions can be formed in a constructive, gradual manner, allowing use of intermediate results.

Proposition 6. Let e be a well-formed expression and e' be a subexpression of e. Then, it holds

$$VC(e') \subseteq VC(e) \text{ and } IC(e') \subseteq IC(e)$$

To see the significance of this proposition, let $\{F_1, ..., F_k\}$ be the facets of a faceted taxonomy and let e' be an expression that defines the current desired compound taxonomy. Assume that now the designer adds some new facets of interest. Then, he has only to extend (and not to rewrite) e' with a subexpression e'' such that the new expression, $e = e' \; op \; e''$, defines the new desired compound taxonomy (see the examples of Appendix B).

The following proposition expresses that the valid and invalid genuine compound terms of a subexpression of an expression e are indeed valid and invalid compound terms, respectively.

Proposition 7. Let e be a well-formed expression and $e_1 \; op \; e_2$ be a subexpression of e. Then, it holds

$$VG(e_1 \; op \; e_2) \subseteq VC(e) \text{ and } IG(e_1 \; op \; e_2) \subseteq IC(e)$$

From the above proposition and propositions 2 and 3, it easily follows that for any parameter P and N of e, it holds: $P \subseteq VC(e)$ and $N \subseteq IC(e)$.

In Section 2, we informally indicated that if a compound term s is valid then every compound term in $\mathrm{Br}(s)$ is also valid. Additionally, if a compound term s is invalid then every compound term in $\mathrm{Nr}(s)$ is also invalid. This property is formally proved in the following proposition.

Proposition 8. Let e be a well-formed expression. It holds:

$$Br(VC(e)) = VC(e) \text{ and } Nr(IC(e)) = IC(e)$$

The semantic interpretation that we described can be extended in a straightforward manner, so as to also capture the plus-self-product operation and the minus-self-product operation.

4 Comparison with Description Logics

In this section we will investigate whether we can represent the compound taxonomies that are defined by CTCA expressions, in Description Logics (DL) [5]. This involves finding a method for representing in DL, taxonomies and the constraints that are imposed by the CTCA expressions, in a way that allows reducing compound term validity checking to the semantics (and inference rules) of DL.

Recall that any Description Logic (DL) is a fragment of First Order Logic (FOL). In particular, any (basic) DL is a subset of the function-free FOL using at most *three* variable names. In DL, a knowledge base, also referred as a DL theory, denoted by Σ, is formed by two components: the *intensional* one, called TBox, (denoted by TB), and the *extensional* one, called ABox (denoted by AB), i.e. $\Sigma = (TB, AB)$. The first is a general schema concerning the classes of individuals to be represented, their general properties and mutual relationships. The latter is a (partial) instantiation of this schema, containing assertions relating either individuals to classes, or individuals to each other. Specifically, the language used is composed by symbols denoting *concept names* and *individual names*[10]. Besides the above symbols, the alphabet includes a number of *constructors* that permit the formation of *concept expressions*. In our case, we only need to use the bottom concept $\bot$ and the conjunctive constructor $\sqcap$. Now a DL knowledge base comprises expressions belonging to one of the following two categories where C, C_1, C_2 stand for concepts, and a for individual constants:

- $C(a)$, called *concept assertions,* asserting that a is an instance of C;
- $C_1 \sqsubseteq C_2$, asserting that C_1 is more specific than C_2.

The set of concept assertions constitute the ABox of Σ, while the latter, which are called *concept axioms*, constitute the TBox of Σ.

The semantics is specified through the notion of interpretation. An *interpretation* $\mathcal{I}$ is a pair $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consisting of a non-empty set $\Delta^{\mathcal{I}}$ (called the *domain*) and of an *interpretation function* $\cdot^{\mathcal{I}}$. The latter maps different individual constants into different elements of $\Delta^{\mathcal{I}}$ and primitive concepts into subsets of $\Delta^{\mathcal{I}}$. The interpretation of complex concepts is defined by structural induction, in our case by the rules: $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$, $\bot^{\mathcal{I}} = \emptyset$ and $(C_1 \sqcap C_2)^{\mathcal{I}} = C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$. Semantically, the assertion $C(a)$ is *satisfied* by an interpretation $\mathcal{I}$ iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$. An axiom $C_1 \sqsubseteq C_2$ is *satisfied* by an interpretation $\mathcal{I}$ iff ${C_1}^{\mathcal{I}} \subseteq {C_2}^{\mathcal{I}}$. An interpretation $\mathcal{I}$ *satisfies* (*is a model of*) a KB Σ iff $\mathcal{I}$ satisfies each axiom in TB and each assertion in AB. A KB Σ *entails* an assertion α (written $\Sigma \models \alpha$) iff every model of Σ satisfies α.

One can easily see that a faceted taxonomy $F = (\mathcal{T}, \leq)$ can be expressed as a TBox containing one primitive concept $\mathtt{t}$ for each term $t \in \mathcal{T}$, and one concept axiom $\mathtt{t} \sqsubseteq \mathtt{t'}$ for each relationship $t \leq t'$ of the taxonomy.

Recall that we have equated compound term validity with non-empty interpretation and compound term invalidity with empty interpretation. One can

[10] We skip *roles* as they are not needed for the problem at hand.

easily see that invalidity reduces quite straightforwardly to unsatisfiability of DL. On the other hand, in order to express that each term t of a taxonomy is valid, we will create an ABox that contains one concept assertion $\mathtt{t}(a_t)$, where a_t is a new individual constant (different terms are associated with different constants, i.e. if $t \neq t'$ then $a_t \neq a_{t'}$). If $(T, \leq)$ is a taxonomy we shall use Σ_T to denote the DL theory that is derived according to the above. For example, if $(T, \leq) = (\{t_1, t_2, t_3\}, \{t_2 \leq t_1, t_3 \leq t_1\})$, then $\Sigma_T = \{\mathtt{t_2} \sqsubseteq \mathtt{t_1}, \mathtt{t_3} \sqsubseteq \mathtt{t_1}, \mathtt{t_1}(1), \mathtt{t_2}(2), \mathtt{t_3}(3)\}$

We will now generalize, and describe how we can construct a DL theory Σ_e for every well-formed expression e of the CTCA[11]. The method is described in Table 2. At first note that a compound term $s = \{t_1, ..., t_n\}$ in the DL framework corresponds to the conjunctively defined concept $d_s = \mathtt{t}_1 \sqcap ... \sqcap \mathtt{t}_n$.

In the case of a plus-product operation, for each $p = \{t_1, ..., t_n\} \in P$ we derive the concept assertion $(\mathtt{t}_1 \sqcap ... \sqcap \mathtt{t}_n)(a_p)$, where a_p is a fresh new constant.

Now in the case of a minus-product operation, for each $\{t_1, ..., t_n\} \in N$ we derive the concept axiom $t_1 \sqcap ... \sqcap t_n \sqsubseteq \bot$.

e	Σ_e
T_i	$\{\ \mathtt{t}(a_t) \mid$ for each $t \in \mathcal{T}_i\} \cup$ $\{\ \mathtt{t} \sqsubseteq \mathtt{t}' \mid$ for each $t, t' \in \mathcal{T}_i$ s.t. $t \leq t'\}$
$\overset{*}{\oplus}_P (T_i)$	$\Sigma_{T_i} \cup \{\ (\mathtt{t}_1 \sqcap ... \sqcap \mathtt{t}_n)(a_p) \mid\ p = \{t_1, ..., t_n\} \in P\}$
$\overset{*}{\ominus}_N (T_i)$	$\Sigma_{T_i} \cup \{\ \mathtt{t}_1 \sqcap ... \sqcap \mathtt{t}_n \dot{\leq} \bot \mid\ \{t_1, ..., t_n\} \in N\}$
$e_1 \oplus_P e_2$	$\Sigma_{e_1} \cup \Sigma_{e_2} \cup \{\ (\mathtt{t}_1 \sqcap ... \sqcap \mathtt{t}_n)(a_p) \mid\ p = \{t_1, ..., t_n\} \in P\}$
$e_1 \ominus_N e_2$	$\Sigma_{e_1} \cup \Sigma_{e_2} \cup \{\ \mathtt{t}_1 \sqcap ... \sqcap \mathtt{t}_n \dot{\leq} \bot \mid\ \{t_1, ..., t_n\} \in N\}$

Table 2. Using DL for representing the compound terminology of a CTCA expression

Having defined Σ_e in this way, Table 3 sketches how we can check whether a compound term $s = \{t_1, ..., t_n\}$ belongs to the compound terminology S_e of an expression e by using Σ_e and the inference mechanisms of DL. In this table we consider that $s_1 = \{t \in s \mid F(t) \in F(e_1)\}$ and $s_2 = \{t \in s \mid F(t) \in F(e_2)\}$, where $F(t)$ is the facet of term t, and $F(e)$ are the facets appearing in e.

Notice the difference between the algorithm for the plus-products with that of the minus-products: if the current operation is a plus-product then validity checking reduces to query answering, while if the current operation is a minus-product then validity checking reduces to satisfiability checking. It follows that if we would like to use a DL-based system for checking the validity of a compound term then we should design a *metasystem* (on top of DL inference engine) that parses the expression e and recursively calls the inference mechanisms of DL (i.e. query answering and concept satisfiability) as described in Table 3. We omit the

[11] It is important that the expression e be well-formed. Otherwise, the TBOX may be inconsistent.

proof that this metasystem would function correctly, because the recursive calls of Table 3 are based exactly on the algorithm $IsValid(e, s)$ as it has been given in [13].

CTCA approach	*A DL-based approach*
$IsValid(T_i, s)$=TRUE	$\{\ a \mid \Sigma_{T_i} \models d_s(a)\} \neq \emptyset$
$IsValid(\overset{*}{\oplus}_P\ (T_i), s)$=TRUE	$\{\ a \mid \Sigma^*_{\oplus_P(T_i)} \models d_s(a)\} \neq \emptyset$
$IsValid(\overset{*}{\ominus}_N\ (T_i), s)$=TRUE	$\Sigma^*_{\ominus_N(T_i)} \not\models d_s \equiv \bot$
$IsValid(e_1 \oplus_P e_2, s)$=TRUE	$(\{\ a \mid \Sigma_{e_1 \oplus_P e_2} \models d_s(a)\} \neq \emptyset)\ \vee$ $IsValid(e_1, s)\ \vee$ $IsValid(e_2, s)$
$IsValid(e_1 \ominus_N e_2, s)$=TRUE	$(\Sigma_{e_1 \ominus_N e_2} \not\models d_s \equiv \bot)\ \wedge$ $IsValid(e_1, s_1)\ \wedge$ $IsValid(e_2, s_2)$, where $s_1 = \{t \in s \mid F(t) \in F(e_1)\}$ and $s_2 = \{t \in s \mid F(t) \in F(e_2)\}$

Table 3. Using DL for checking the validity of a compound term

Alternatively, if we want to use the classical reasoning services of DL, then we cannot create the Σ_e by the method described in Table 2. Instead, we have to either:
(a) convert all minus-products to plus-products (and then translate the resulting plus-products to DL), or
(b) convert all plus-products to minus-products (and then translate the resulting minus-products to DL).

We can convert a minus-product operation $e_1 \ominus_N e_2$ to a plus-product operation (and vice-versa) as follows:

$$e_1 \ominus_N e_2 = e_1 \oplus_P e_2 \text{ where } P = G_{S_{e_1}, S_{e_2}} - Nr(N)$$
$$e_1 \oplus_P e_2 = e_1 \ominus_N e_2 \text{ where } N = G_{S_{e_1}, S_{e_2}} - Br(P)$$

So according to approach (a), we translate each plus-product operation as described in Table 2, and each minus-product operation $e_1 \ominus_N e_2$ as:

$$\Sigma_{e_1} \cup \Sigma_{e_2} \cup \{\ d_s(a_s) \mid \quad s \in G_{S_{e_1}, S_{e_2}} - Nr(N)\}$$

It is evident that if we derive Σ_e in this way, it holds:

$$s \in S_e \text{ iff } \{\ a \mid \Sigma_e \models d_s(a)\} \neq \emptyset$$

Now according to approach (b) we translate each minus-product operation as described in Table 2, and each plus-product operation $e_1 \oplus_P e_2$ as follows:

$$\Sigma_{e_1} \cup \Sigma_{e_2} \cup \{\ d_s \sqsubseteq \bot \mid \quad s \in G_{S_{e_1}, S_{e_2}} - Br(P)\}$$

It is again evident that if we derive Σ_e in this way, it holds:

$$s \in S_e \text{ iff } \Sigma_e \not\models d_s \equiv \bot$$

However note, that in both (a) and (b) approaches, the conversion of plus-products to minus-products (or the reverse) requires computing $G_{S_{e_1},S_{e_2}}$ which in turn requires computing S_{e_1} and S_{e_2}. This might turn out computationally heavy. Recall that the reason that CTCA supports both positive and negative statements (i.e. plus-products and minus-products) is to allow the designer to select at each step the most economical operation i.e. the one that requires providing the less number of parameters. Under this assumption, it follows that the above conversion is expected to result in an expression with much more parameters, i.e. to a much bigger in size DL theory.

From the above discussion it is evident that we cannot represent CTCA expressions in DL in a straightforward manner (due to the closed-world assumptions inherent to the operations of CTCA). In addition, in the DL framework there is no clear method for deciding whether an expression is well-formed.

5 Conclusion

In this paper, we defined the semantics of the Compound Term Composition Algebra (CTCA). Specifically, we justified the definition of the algebraic operations, based on the model-theoretic definition of the valid and invalid genuine compound terms. Having defined the valid (resp. invalid) genuine compound terms of a positive (resp. negative) operation, the invalid (resp. valid) genuine compound terms are computed based on a closed-world assumption. The valid compound terms according to an expression e is the union of the valid genuine compound terms of all operations of e.

Additionally, we defined the models of an algebraic expression. Intuitively, a model of an algebraic expression, is an interpretation which is non-empty for each valid compound term, and empty for each invalid compound term. We proved that every well-formed algebraic expression is satisfiable. Moreover, we proved that well-formed algebraic expressions are monotonic, which ensures that results of subexpressions are not invalidated by larger expressions. We also showed that we cannot directly represent the compound taxonomies defined by CTCA directly in Description Logics, and a metasystem was designed on top of Description Logics to implement the algebra.

CTCA can be used in any application that indexes objects using a faceted taxonomy. For example, it can be used for designing compound taxonomies for products, for fields of knowledge (e.g. indexing the books of a library), etc.

As we can infer the valid compound terms of a faceted taxonomy, we are able to generate a single hierarchical navigation tree *on the fly*, having only valid compound terms as nodes. The algorithm for deriving navigation trees on the fly is given in [13]. Such a navigational tree can be used for object indexing,

preventing indexing errors, as well as for object retrieval, guiding the user to only meaningful selections.

Moreover, CTCA can be used for providing compact representations. This is because, there is no need to store the complete set of valid compound terms of a faceted taxonomy. Only the faceted taxonomy and the expression have to be stored. A novel application of CTCA for compressing Symbolic Data Tables is described in [15]. For more about Symbolic Data Analysis, see [3, 4].

The algebra can also be used for query answering optimization. For example, consider Figure 1, and assume that the user wants to retrieve all hotels located in Greece and offer winter sports. As $\{Islands, WinterSports\}$ is an invalid compound term, the system (optimizing execution) does not have to look for hotels located in islands at all.

Another application of the algebra is consistency control. In certain applications, objects may be indexed to non-meaningful compound terms (due to lack of information or other factors). In such case, the algebra can help to point-out the incorrectly indexed objects. Genomic experiments belong to this category, as several aspects of the genomic domain are still unknown, and experimental methods may be inaccurate or based on erroneous assumptions. The *Gene Ontology* (GO)[12] is a faceted taxonomy with 3 facets, namely *Molecular Function, Biological Process*, and *Cellular Component*. Genes may be indexed to one or more terms of each facet. The annotation guide of GO indicates that indexing of genes to contradictory compound terms is allowed, as long as indexing of a gene by a term is associated with the type of evidence and a cited source. Specifying the valid compound terms of GO using the algebra, genes indexed by an invalid compound term, can be immediately designated by an inconsistency flag. Certainly, knowing the genes indexed by an invalid compound term is of interest to biologists who need to perform more elaborate experiments to correct inconsistencies.

The algebra can also be used for configuration management. Consider a product whose configuration is determined by a number of parameters, each associated with a finite number of values. However, some configurations may be unsupported, unviable, or unsafe. For this purpose, the product designer can employ an expression which specifies all valid configurations, thus ensuring that the user selects only among these.

As future work, we plan to study how updates on the faceted taxonomy, or changes to the desired compound terminology should update the expression that defines the compound terminology. This process can be automated. This is very important in practice, as it adds flexibility to the design process: the designer during the formulation of the expression e can update the faceted taxonomy, without having to bother that e will become obsolete. Additionally, the designer can add or delete compound terms from the desired compound terminology without having to worry that e will no longer reflect his/her desire.

[12] http://www.geneontology.org/

References

1. "XFML: eXchangeable Faceted Metadata Language". http://www.xfml.org.
2. "XFML+CAMEL:Compound term composition Algebraically-Motivated Expression Language". http://www.csi.forth.gr/markup/xfml+camel.
3. H. H. Bock and E. Diday. *Analysis of Symbolic Data*. Springer-Verlag, 2000. ISBN: 3-540-66619-2.
4. Edwin Diday. "An Introduction to Symbolic Data Analysis and the Sodas Software". *Journal of Symbolic Data Analysis*, 0(0), 2002. ISSN 1723-5081.
5. F.M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. "Reasoning in Description Logics". In Gerhard Brewka, editor, *Principles of Knowledge Representation*, chapter 1, pages 191–236. CSLI Publications, 1996.
6. Elizabeth B. Duncan. "A Faceted Approach to Hypertext". In Ray McAleese, editor, *HYPERTEXT: theory into practice, BSP*, pages 157–163, 1989.
7. P. H. Lindsay and D. A. Norman. *Human Information Processing*. Academic press, New York, 1977.
8. Amanda Maple. "Faceted Access: A Review of the Literature", 1995. http://theme.music.indiana.edu/tech_s/mla/facacc.rev.
9. Ruben Prieto-Diaz. "Classification of Reusable Modules". In *Software Reusability. Volume I*, chapter 4, pages 99–123. acm press, 1989.
10. Ruben Prieto-Diaz. "Implementing Faceted Classification for Software Reuse". *Communications of the ACM*, 34(5):88–97, 1991.
11. U. Priss and E. Jacob. "Utilizing Faceted Structures for Information Systems Design". In *Proceedings of the ASIS Annual Conf. on Knowledge: Creation, Organization, and Use (ASIS'99)*, October 1999.
12. S. R. Ranganathan. "The Colon Classification". In Susan Artandi, editor, *Vol IV of the Rutgers Series on Systems for the Intellectual Organization of Information*. New Brunswick, NJ: Graduate School of Library Science, Rutgers University, 1965.
13. Y. Tzitzikas, A. Analyti, N. Spyratos, and P. Constantopoulos. "An Algebraic Approach for Specifying Compound Terms in Faceted Taxonomies". In *Information Modelling and Knowledge Bases XV, 13th European-Japanese Conference on Information Modelling and Knowledge Bases, EJC'03*, pages 67–87. IOS Press, 2004.
14. Y. Tzitzikas, R. Launonen, M. Hakkarainen, P. Kohonen, T. Leppanen, E. Simpanen, H. Tornroos, P. Uusitalo, and P. Vanska. "FASTAXON: A system for FAST (and Faceted) TAXONomy design". In *Procs. of 23th Int. Conf. on Conceptual Modeling, ER'2004*, Shanghai, China, November 2004. (an on-line demo is available at http://fastaxon.erve.vtt.fi/).
15. Yannis Tzitzikas. "An Algebraic Method for Compressing Very Large Symbolic Data Tables". In *Procs. of the Workshop on Symbolic and Spatial Data Analysis of ECML/PKDD 2004*, Pisa, Italy, September 2004.
16. Yannis Tzitzikas and Anastasia Analyti. "Mining the Meaningful Compound Terms from Materialized Faceted Taxonomies ". In *Procs. of the 3rd Intern. Conference on Ontologies, Databases and Applications of Semantics, ODBASE'2004*, pages 873–890, Larnaca, Cyprus, October 2004.
17. Yannis T. Tzitzikas. *"Collaborative Ontology-based Information Indexing and Retrieval"*. PhD thesis, Department of Computer Science - University of Crete, September 2002.
18. B. C. Vickery. "Knowledge Representation: A Brief Review". *Journal of Documentation*, 42(3):145–159, 1986.

19. W. A. Woods. "Understanding Subsumption and Taxonomy". In *Principles of Semantic Networks*, chapter 1. Morgan Kaufmann Publishers, 1991.

Appendix A: Proofs

In this Appendix, we give the proofs of the propositions appearing in the paper.

We first prove an auxiliary lemma.

Lemma 1 Let S be the compound terminology of an algebraic expression. It holds $Br(S) = S$.
Proof:
We will prove Lemma 1, recursively. Obviously, it holds $S \subseteq Br(S)$.
Let $S = T_i$ then from the definition of a basic compound taxonomy, it follows $Br(T_i) = T_i$.
Let $S = \overset{*}{\oplus}_P T_i$, then $Br(S) = Br(T_i \cup Br(P)) = Br(T_i) \cup Br(Br(P)) = T_i \cup Br(P) = S$.
Let $S = \overset{*}{\ominus}_N T_i$, then $Br(S) = Br(\overset{*}{\oplus} T_i - Nr(N))$. We will show $Br(\overset{*}{\oplus} T_i - Nr(N)) \subseteq \overset{*}{\oplus} T_i - Nr(N)$. Let $s \in Br(\overset{*}{\oplus} T_i - Nr(N))$. Then, there is $s' \in \overset{*}{\oplus} T_i - Nr(N)$ such that $s' \preceq s$. If $s \in Nr(N)$ then $s' \in Nr(N)$, which is impossible. Thus, $s \in \overset{*}{\oplus} T_i - Nr(N)$. Therefore, $Br(S) = S$.
Let $S = \oplus_P(S_1, ..., S_n)$, such that $Br(S_i) = S_i$, for $i = 1, ..., n$. It holds $Br(S) = Br(S_1) \cup ... \cup Br(S_n) \cup Br(Br(P)) = S_1 \cup ... \cup S_n \cup Br(P) = S$.

Let $S = S_1 \oplus ... \oplus S_n$ such that $Br(S_i) = S_i$, for $i = 1, ..., n$. We will show $Br(S) = S$. Let $s \in Br(S_1 \oplus ... \oplus S_n)$. Then, $\exists s' \in S_1 \oplus ... \oplus S_n$ such that $s' \leq s$. Let $s' = s'_1 \cup ... \cup s'_n$, where $s'_i \in S_i$, for $i = 1, ..., n$. Thus, $s = s_1 \cup ... \cup s_n$, where $s_i \in S_i$ and $s'_i \leq s_i$, for $i = 1, ..., n$. Thus, $s_i \in Br(S_i)$, for $i = 1, ..., n$. Therefore, $s \in Br(S_1) \oplus ... \oplus Br(S_n) = S_1 \oplus ... \oplus S_n$.

Let $S = \ominus_N(S_1, ..., S_n)$ such that $Br(S_i) = S_i$, for $i = 1, ..., n$. It holds $Br(S) = Br(S_1 \oplus ... \oplus S_n - Nr(N))$. We will show $Br(S_1 \oplus ... \oplus S_n - Nr(N)) \subseteq S_1 \oplus ... \oplus S_n - Nr(N)$. Let $s \in Br(S_1 \oplus ... \oplus S_n - Nr(N))$. Then, there is $s' \in S_1 \oplus ... \oplus S_n - Nr(N)$ such that $s' \preceq s$. If $s \in Nr(N)$ then $s' \in Nr(N)$, which is impossible. Thus, $s \in Br(S_1 \oplus ... \oplus S_n) - Nr(N) = S_1 \oplus ... \oplus S_n - Nr(N)$. Therefore, $Br(S) = S$. $\diamond$

Proposition 1 Let S be a compound taxonomy over a taxonomy T, and let s and s' be two elements of S. It holds:

$$s \preceq s' \text{ iff } I(s) \subseteq I(s') \text{ in every model } I \text{ of } (T, \leq)$$

Proof:
($\Rightarrow$)

Let $s = \{a_1, ..., a_m\}$ and $s' = \{b_1, ..., b_n\}$. If $s \preceq s'$, then for each b_i exists a_j such that $a_j \leq b_i$. This means that in every model I of $(T, \leq)$, it holds $I(a_j) \subseteq I(b_i)$. Now, as $I(s) = I(a_1) \cap ... \cap I(a_k)$ and $I(s') = I(b_1) \cap ... \cap I(b_m)$, it is evident that it holds $I(s) \subseteq I(s')$ in every model I of $(T, \leq)$.
($\Leftarrow$)

Let $s = \{a_1, ..., a_m\}$ and $s' = \{b_1, ..., b_n\}$. As it has been shown in [17], $I(s) \subseteq I(s')$ in every model of $(T, \leq)$ iff $r(s \cup s') = r(s)$, where $r(\{t_1, ..., t_n\}) = minimal_{\leq}(\{c(t_1), ..., c(t_k)\})$

where $c(t)$ denotes the equivalence class of a term t. However, $r(s \cup s') = r(s)$ can hold only if for each $t' \in s'$ exists $t \in s$ such that $t \leq t'$. Thus it must hold $s \preceq s'$. $\diamond$

Proposition 2 $VG(e \oplus_P e') = Br(P) \cap G_{VC(e),VC(e')}$
Proof:
First, we will show that $VG(e \oplus_P e') \subseteq Br(P) \cap G_{VC(e),VC(e')}$. Let $s \in VG(e \oplus_P e')$. It holds $I(s) \neq \emptyset$, in every model I such that $I(p) \neq \emptyset$, $\forall p \in P$. Therefore, it holds that exists $p \in P$ such that $I(p) \subseteq I(s)$, for every model I. From Prop. 1, it follows that $p \preceq s$. Therefore, $s \in Br(P)$. As $VG(e \oplus_P e') \subseteq G_{VC(e),VC(e')}$, it follows that $s \in Br(P) \cap G_{VC(e),VC(e')}$.

We will now show that $Br(P) \cap G_{VC(e),VC(e')} \subseteq VG(e \oplus_P e')$. Let $s \in Br(P) \cap G_{VC(e),VC(e')}$. Thus, there is $p \in P$ such that $p \preceq s$. From Prop. 1, it follows that $I(p) \subseteq I(s)$, for every model I. Thus, $s \in VG(e \oplus_P e')$. $\diamond$

Proposition 3 $IG(e \ominus_N e') = Nr(N) \cap G_{VC(e),VC(e')}$
Proof:
First, we will show that $IG(e \ominus_N e') \subseteq Nr(N) \cap G_{VC(e),VC(e')}$. Let $s \in IG(e \ominus_N e')$. It holds $I(s) = \emptyset$, in every model I such that $I(n) = \emptyset$, $\forall n \in N$. Therefore, it holds that exists $n \in N$ such that $I(s) \subseteq I(n)$, for every model I. From Prop. 1, it follows that $s \preceq n$. Therefore, $s \in Nr(N)$. As $IG(e \ominus_N e') \subseteq G_{VC(e),VC(e')}$, it follows that $s \in Nr(N) \cap G_{VC(e),VC(e')}$.

We will now show that $Nr(N) \cap G_{VC(e),VC(e')} \subseteq IG(e \ominus_N e')$. Let $s \in Nr(N) \cap G_{VC(e),VC(e')}$. Thus, there is $n \in N$ such that $s \preceq n$. From Prop. 1, it follows that $I(s) \subseteq I(n)$, for every model I of $(\mathcal{T}, \leq)$. Thus, $s \in IG(e \ominus_N e')$. $\diamond$

Proposition 4 Let e be a well-formed expression. There always exists a model I of $(\mathcal{T}, \leq)$ that satisfies e.
Proof:
We create a model I of $(\mathcal{T}, \leq)$ as follows:
Initially, $I(t) = \emptyset$, for every $t \in \mathcal{T}$. Let $VC(e) = \{s_1, ..., s_n\}$. For each $s_i = \{t_{i,1}, ..., t_{i,n_i}\} \in VC(e)$, we insert an object o_i to each $I(t_{i,j})$, for $i = 1, ...n$ and $j = 1, ..., n_i$.
For each $t \leq t'$, we extend $I(t')$ such that $I(t) \subseteq I(t')$. For each $t \in \mathcal{T}$, $I(t)$ contains nothing else.

From the construction, I is a model of $(\mathcal{T}, \leq)$. Additionally, $\forall s \in VC(e)$, obviously, it holds that $I(s) \neq \emptyset$. We will show that $\forall s \in IC(e)$, it holds that $I(s) = \emptyset$. Assume that $\exists s \in IC(e)$ such that $I(s) \neq \emptyset$. Then, there should be an $s' \in VC(e)$ such that $s' \preceq s$. From Prop. 5 and Lemma 1. it follows that $Br(VC(e)) = VC(e)$. Thus, $s \in VC(e)$, which is impossible as $VC(e) \cap IC(e) = \emptyset$. Therefore, $\forall s \in IC(e)$, it holds that $I(s) = \emptyset$. $\diamond$

Proposition 5 Let e be a well-formed expression. It holds:

$$VC(e) = S_e \text{ and } IC(e) = P(\mathcal{T}_e) - S_e$$

Proof:
We will prove the proposition recursively.
The proposition obviously holds for $e = T_i$.

Let $e = e_1 \oplus_P e_2$, and assume that the proposition holds for e_1 and e_2.
Using Proposition 2 and assumption, we have: $VC(e_1 \oplus_P e_2) = VG(e_1 \oplus_P e_2) \cup VC(e_1) \cup VC(e_2) = (Br(P) \cap G_{S_{e_1}, S_{e_2}}) \cup S_{e_1} \cup S_{e_2} = (Br(P) \cap (S_{e_1} \oplus S_{e_2})) \cup S_{e_1} \cup S_{e_2}$. As e is well-formed, it holds $Br(P) \subseteq S_{e_1} \oplus S_{e_2}$. Therefore, $VC(e_1 \oplus_P e_2) = Br(P) \cup S_{e_1} \cup S_{e_2} = S_e$.

Now, let $e = e_1 \ominus_N e_2$, and assume that the proposition holds for e_1 and e_2.
Using Proposition 3 and assumption, we have: $VC(e_1 \ominus_N e_2) = VG(e_1 \ominus_N e_2) \cup VC(e_1) \cup VC(e_2) = (G_{S_{e_1}, S_{e_2}} - Nr(N)) \cup S_{e_1} \cup S_{e_2}$. As e is well-formed, it holds $Nr(N) \cap S_{e_1} = \emptyset$ and $Nr(N) \cap S_{e_2} = \emptyset$. Therefore, $VC(e_1 \ominus_N e_2) = S_{e_1} \oplus S_{e_2} - Nr(N) = S_e$.

Therefore, for any expression e, $VC(e) = S_e$. Now, it follows immediately that $IC(e) = P(\mathcal{T}_e) - S_e$. $\diamond$

Proposition 6 Let e be a well-formed expression and e' be a subexpression of e. Then, it holds

$$VC(e') \subseteq VC(e) \text{ and } IC(e') \subseteq IC(e)$$

Proof:
The fact that $VC(e') \subseteq VC(e)$ follows recursively from the definition of $VC(e)$.
We will now show that $IC(e') \subseteq IC(e)$.

Let $e = e_1 \ op \ e_2$. Then, $IC(e) = P(\mathcal{T}_e) - VC(e) = P(\mathcal{T}_e) - VC(e_1 \ op \ e_2) = P(\mathcal{T}_e) - VG(e_1 \ op \ e_2) - VC(e_1) - VC(e_2) \supseteq P(\mathcal{T}_{e_1}) - VG(e_1 \ op \ e_2) - VC(e_1) - VC(e_2)$. As $VG(e_1 \ op \ e_2) \cap P(\mathcal{T}_{e_1}) = \emptyset$, and $VC(e_2) \cap P(\mathcal{T}_{e_1}) = \emptyset$, it holds that $IC(e) \supseteq P(\mathcal{T}_{e_1}) - VC(e_1) = IC(e_1)$. Similarly, $IC(e_2) \subseteq IC(e)$.

Recursively, it holds that for any subexpression e' of e, it holds that $IC(e') \subseteq IC(e)$. $\diamond$

Proposition 7 Let e be a well-formed expression and $e_1 \ op \ e_2$ be a subexpression of e. Then, it holds

$$VG(e_1 \ op \ e_2) \subseteq VC(e) \text{ and } IG(e_1 \ op \ e_2) \subseteq IC(e)$$

Proof:
The fact that $VG(e_1 \ op \ e_2) \subseteq VC(e)$ follows recursively from the definition of $VC(e)$.
We will now show that $IG(e_1 \ op \ e_2) \subseteq IC(e)$.

From Prop. 6, it holds that $IC(e_1 \ op \ e_2) \subseteq IC(e)$. Now, $IC(e_1 \ op \ e_2) = P(\mathcal{T}_{e_1 \ op \ e_2}) - VG(e_1 \ op \ e_2) - VC(e_1) - VC(e_2) \supseteq IG(e_1 \ op \ e_2)$. Therefore, $IG(e_1 \ op \ e_2) \subseteq IC(e)$. $\diamond$

Proposition 8 Let e be a well-formed expression. It holds:

$$Br(VC(e)) = VC(e) \text{ and } Nr(IC(e)) = IC(e)$$

Proof:
For Lemma 1 and Proposition 5, it follows immediately that $Br(VC(e)) = VC(e)$.

Obviously, $IC(e) \subseteq Nr(IC(e))$. We will prove that $Nr(IC(e)) \subseteq IC(e)$. Let $s \in Nr(IC(e))$. Then, there is $s' \in IC(e)$ such that $s \preceq s'$. For the definition of $IC(e)$, it follows that $s' \in P(\mathcal{T}_e) - VC(e)$. Assume that $s \notin IC(e)$. Then, $s \in VC(e)$, which implies that $s' \in VC(e)$. However, this is impossible. Thus, $s \in IC(e)$. $\diamond$

Appendix B: Example

Suppose that the domain of interest is a set of hotel Web pages and that we want to index these pages using a faceted taxonomy. First, we must define the taxonomy. Suppose it is decided to do the indexing according to three facets, namely the *location* of the hotels, the kind of *accommodation*, and the *facilities* they offer. Specifically, assume that the designer employs (or designs from scratch) the facets shown in Figure 4.

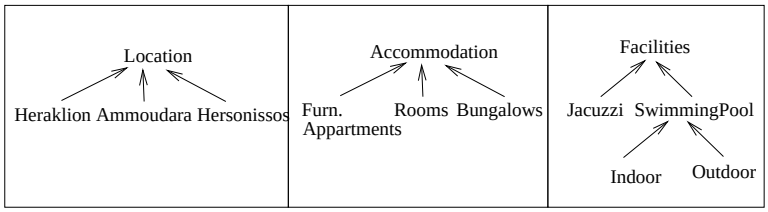

Fig. 4. Three-facets

The faceted taxonomy has 13 terms ($|\mathcal{T}|$=13) and $P(\mathcal{T})$ has 890 compound terms[13]. However, available domain knowledge suggests that only 96 compound terms are valid. Omitting the compound terms which are singletons or contain top terms of the facets, the following 23 valid compound terms remain:

$\{Heraklion, Furn.Appartments, \}, \{Heraklion, Rooms\},$
$\{Ammoudara, Furn.Appartments\}, \{Ammoudara, Rooms\},$
$\{Ammoudara, Bungalows\}, \{Hersonissos, Furn.Appartments\},$
$\{Hersonissos, Rooms\}, \{Hersonissos, Bungalows\},$
$\{Hersonissos, SwimmingPool\}, \{Hersonissos, Indoor\},$
$\{Hersonissos, Outdoor\}, \{Ammoudara, Jacuzzi\},$
$\{Rooms, SwimmingPool\}, \{Rooms, Indoor\},$
$\{Bungalows, SwimmingPool\}, \{Bungalows, Outdoor\},$
$\{Bungalows, Jacuzzi\}, \{Hersonissos, Rooms, SwimmingPool\},$
$\{Hersonissos, Rooms, Indoor\}, \{Hersonissos, Bungalows, SwimmingPool\},$
$\{Hersonissos, Bungalows, Outdoor\}, \{Ammoudara, Bungalows, Jacuzzi\}.$

Rather than being explicitly enumerated, the 96 valid compound terms can be algebraically specified. In this way, the specification of the desired compound terms can

[13] Equivalent compound terms are considered the same. Thus, $|P(\mathcal{T})|$ is not 2^{13} but 890. This is computed as follows: It holds that $|\overset{*}{\oplus}(Location)|$=8, $|\overset{*}{\oplus}(Accomodation)| = 8$, and $|\overset{*}{\oplus}(Facilities)| = 10$. Thus, $|P(\mathcal{T})| = |(\overset{*}{\oplus}(Location)) \oplus (\overset{*}{\oplus}(Accomodation)) \oplus (\overset{*}{\oplus}(Facilities)|| = (8 + 8 * 8 + 8 * 10 + 8 * 8 * 10) + (8 + 8 * 10) + 10 = 890$.

be done in a systematic, gradual, and easy manner. For example, the following plus-product operation can be used:

$$\oplus_P(Location, Accommodation, Facilities), \text{ where}$$

$$\begin{aligned} P = \ & \{\{Heraklion, Furn.Appartments\}, \\ & \{Heraklion, Rooms\}, \\ & \{Ammoudara, Furn.Appartments\}, \\ & \{Ammoudara, Rooms\}, \\ & \{Hersonissos, Furn.Appartments\}, \\ & \{Ammoudara, Bungalows, Jacuzzi\}, \\ & \{Hersonissos, Rooms, Indoor\}, \\ & \{Hersonissos, Bungalows, Outdoor\}\} \end{aligned}$$

Note that the compound terms in P are only 8. Alternatively, the same result can be obtained more efficiently through the expression:

$$(Location \ominus_N Accommodation) \oplus_P Facilities,$$
where

$$\begin{aligned} N = \ & \{\{Heraklion, Bungalows\}\}, \text{ and} \\ P = \ & \{\{Hersonissos, Rooms, Indoor\}, \\ & \{Hersonissos, Bungalows, Outdoor\}, \\ & \{Ammoudara, Bungalows, Jacuzzi\}\} \end{aligned}$$

Note that now the total number of compound terms in P and N is just 4. In summary, the faceted taxonomy of our example, includes 13 terms, 890 compound terms, and 96 valid compound terms which can be specified by providing only 4 (carefully selected) compound terms and an appropriate algebraic expression.

Consider now the additional facet *Season* shown in Figure 5, and suppose that $\{Bungalows, Winter\}$ is an invalid combination of compound terms between the previous compound taxonomy $(Location \ominus_N Accommodation) \oplus_P Facilities$ and the basic compound taxonomy *Season*.

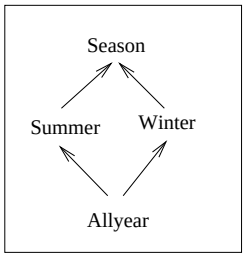

Fig. 5. The facet *Season*

Then, the designer can declare the expression

$$((Location \ominus_N Accommodation) \oplus_P Facilities) \ominus_{N'} Season$$

where $N' = \{\{Bungalows, Winter\}\}$

Note that the total number of compound terms in N, P, and N' is 5. The number of valid compound terms is 530. Note that the new expression is well-formed. Thus, previous results are not invalidated.

The same result could be obtained through the less efficient operation

$$\oplus_{P'}(Location, Accommodation, Facilities, Season), \text{ where}$$

$$\begin{aligned} P' = \ & \{\{Heraklion, Furn.Appartments, Allyear\}, \{Heraklion, Rooms, Allyear\}, \\ & \{Ammoudara, Furn.Appartments, Allyear\}, \{Ammoudara, Rooms, Allyear\}, \\ & \{Hersonissos, Furn.Appartments, Allyear\}, \{Ammoudara, Bungalows, Jacuzzi, Summer\}, \\ & \{Hersonissos, Rooms, Indoor, Allyear\}, \{Hersonissos, Bungalows, Outdoor, Summer\}\} \end{aligned}$$

In this case the number of compound terms in P' is 8.

We will now give an example of an expression which includes a *minus-self-product* operation. Consider the faceted taxonomy of Figure 6.

Fig. 6. Another faceted taxonomy

The user can declare the expression:

$$Location \oplus_P (\overset{*}{\ominus}_N (Sports)), \text{ where}$$

$$\begin{aligned} N = \ & \{\{SeaSports, WinterSports\}\}, \text{ and} \\ P = \ & \{\{Heraklion, SeaSki, Windsurfing\}, \{Olymbus, SnowSki\}\} \end{aligned}$$

Appendix C: Table of Symbols

Symbol	Definition
$P(.)$	$Powerset$
$s \preceq s'$	$\forall t' \in s' \;\; \exists t \in s \;\;$ such that $\;\; t \leq t'$
T_i	$\cup\{\ \mathrm{Br}(\{t\}) \mid t \in \mathcal{T}_i\}$
$S_1 \oplus ... \oplus S_n$	$\{\ s_1 \cup ... \cup s_n \mid s_i \in S_i\}$
$\oplus_P(S_1, ...S_n)$	$S_1 \cup ... \cup S_n \;\cup\; Br(P)$
$\ominus_N(S_1, ...S_n)$	$S_1 \oplus ... \oplus S_n - Nr(N)$
$\overset{*}{\oplus}(T_i)$	$P(\mathcal{T}_i)$
$\overset{*}{\oplus}_P(T_i)$	$T_i \cup Br(P)$
$\overset{*}{\ominus}_N(T_i)$	$\overset{*}{\oplus}(T_i) - Nr(N)$
$G_{S_1,...,S_n}$	$S_1 \oplus ... \oplus S_n - \cup_{i=1}^{n} S_i$
G_{T_i}	$\overset{*}{\oplus}(T_i) - T_i$
S_e	the evaluation of an expression e
$\hat{I}(\{t_1, ..., t_n\})$	$I(t_1) \cap ... \cap I(t_n)$
$d_{\{t_1,...,t_n\}}$	$\mathtt{t}_1 \sqcap ... \sqcap \mathtt{t}_n$
$F(t)$	the facet of term t
$F(e)$	the facets that appear in expression e

Table 4. Table of Symbols